SQL Server 2000 : Relational Engine Enhancements

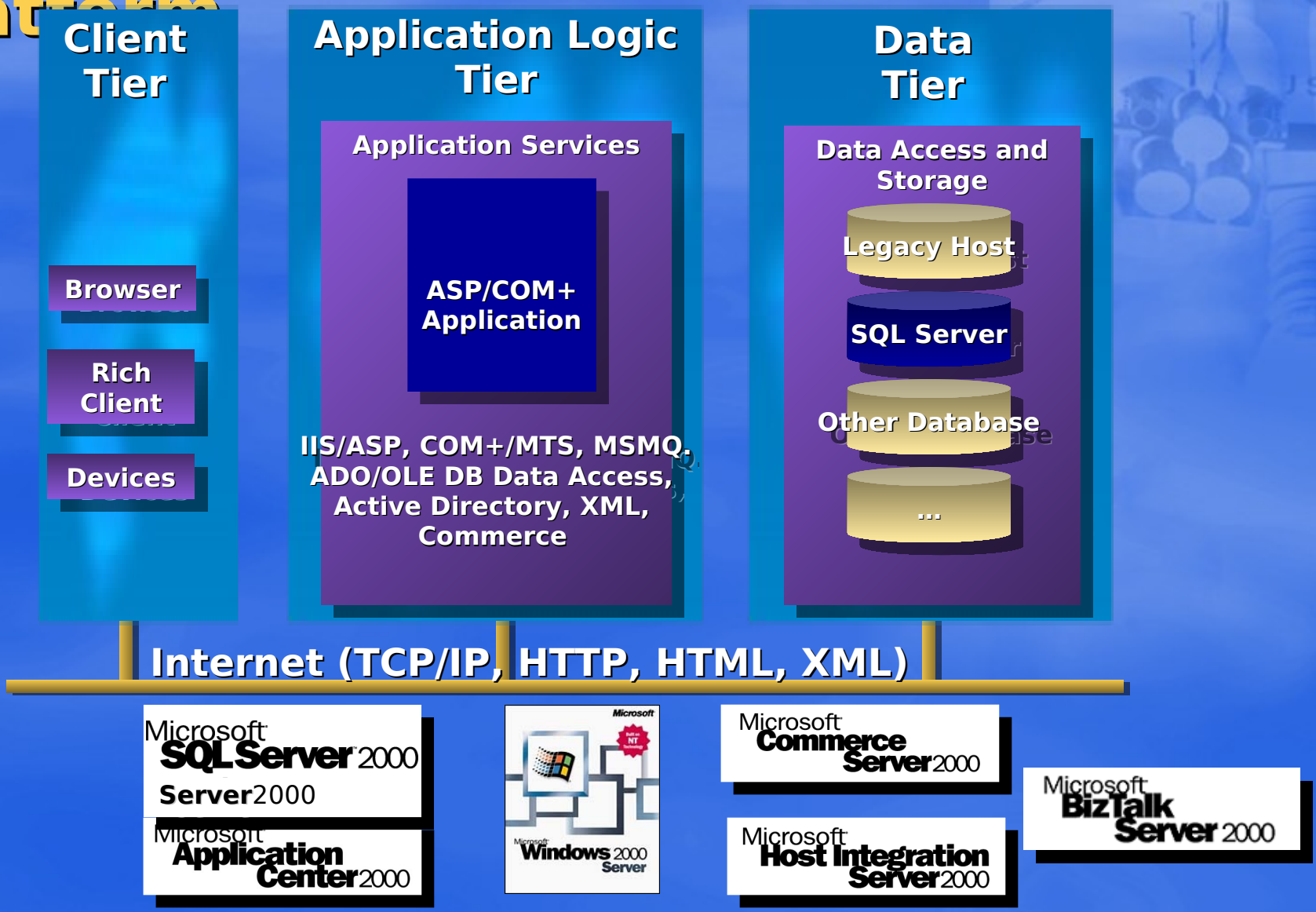
James Hamilton, Development Manager
JamesRH@microsoft.com

Microsoft Corporation

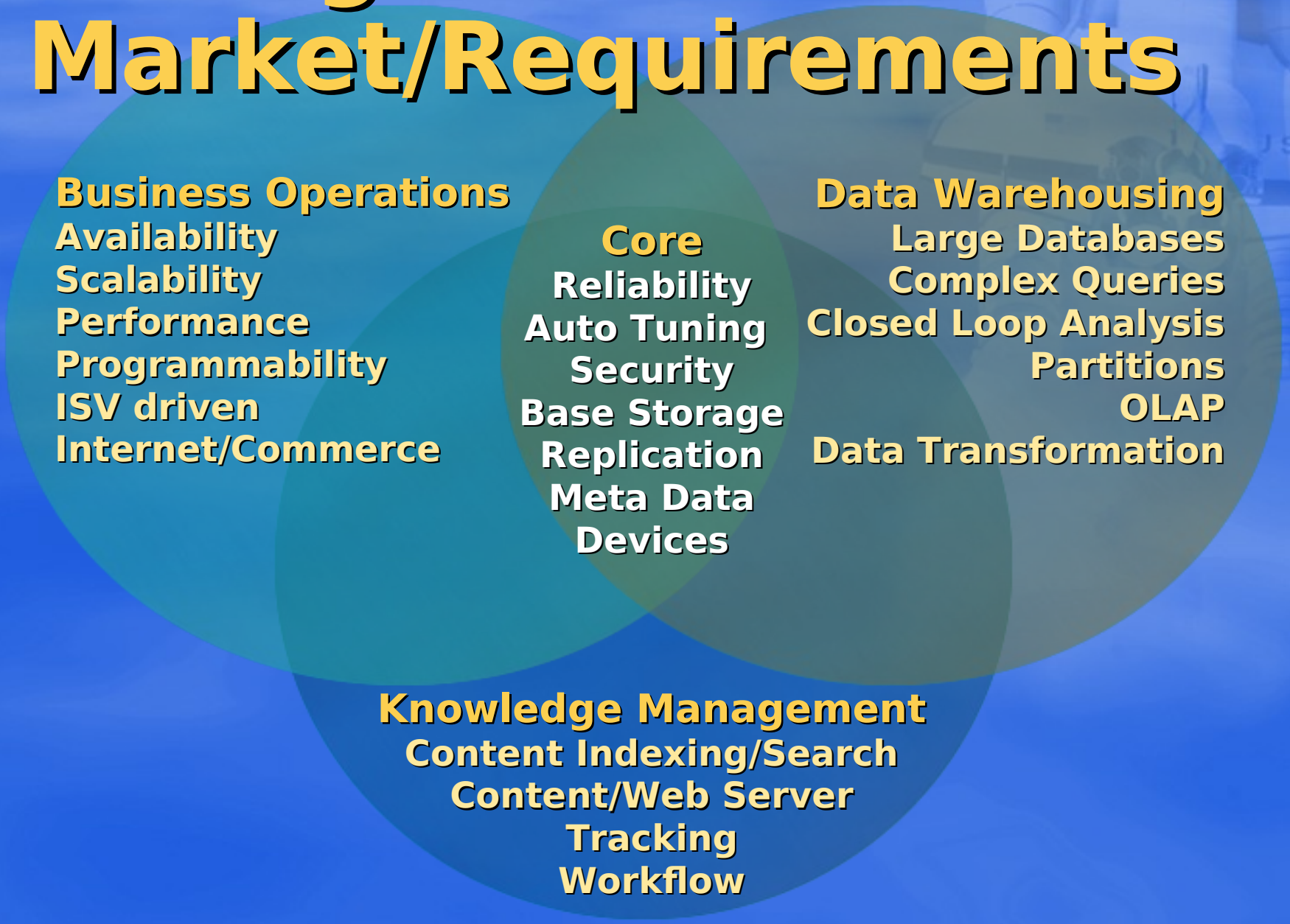

POWER UP
Windows DNA 2000
Readiness Conference
featuring SQL Server 2000

Windows DNA 2000

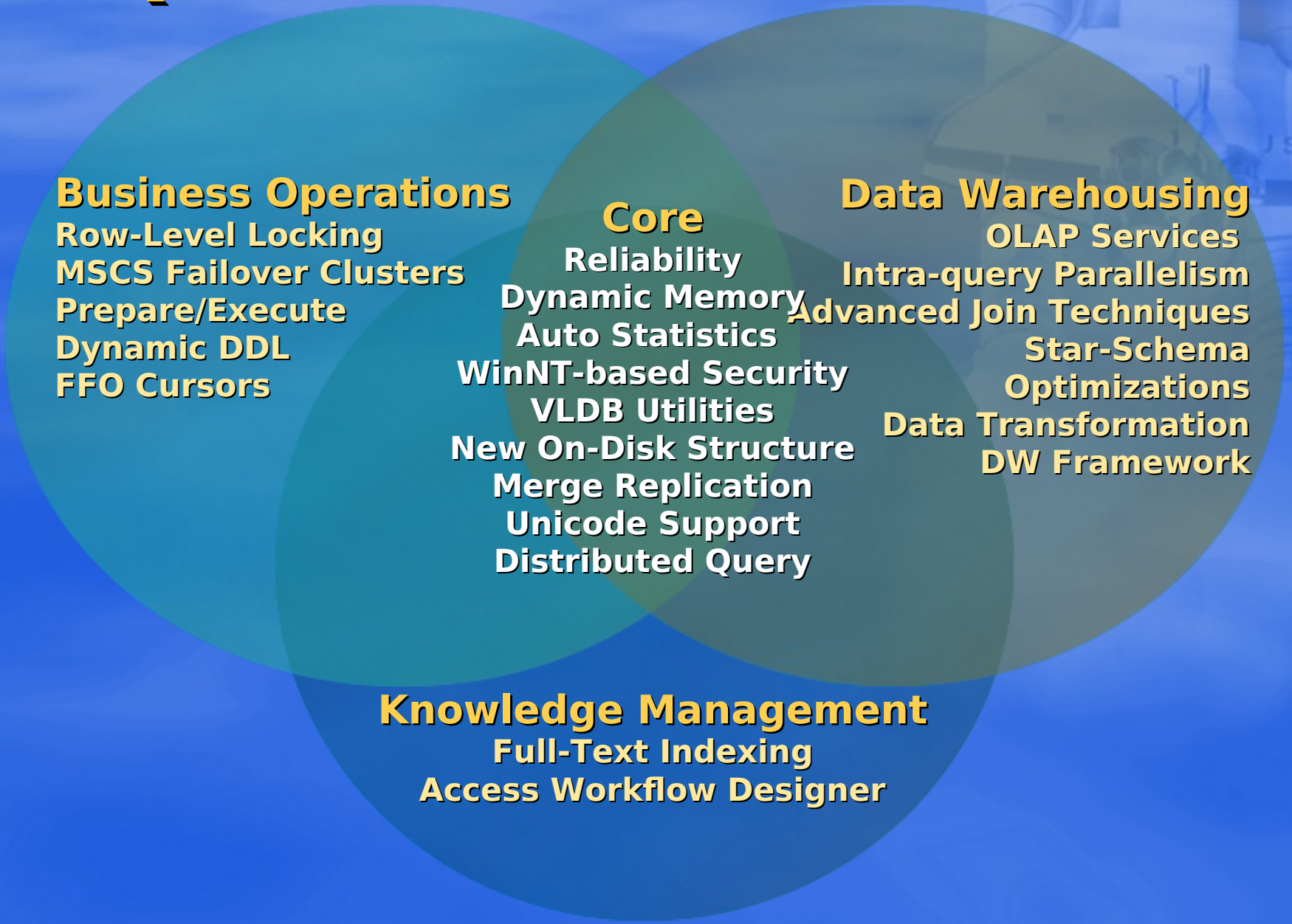
Next Generation Web Application Platform



Storage Market/Requirements



SQL Server 7.0



SQL Server 7.0 : Relational Engine

- State-of-the-art OLTP & DSS QP
- Intra-query parallelism
- Advanced join technology
- Caching & auto-parameterization of query plans
- Unicode
- Distributed heterogeneous queries
- Full-Text Search

SQL Server 2000: Relational Engine

- **Programmability**
- **XML support**
- **Manageability**
  - **Multiple SQL Server instances**
  - **Security**
- **Scalability & performance**
- **Availability**

Programmability

- Cascaded DRI
- User defined functions
- Column level collations
- Instead of triggers
- New datatypes
- Full text index enhancements
- Extended properties
- Checksum
- Session context

Cascaded DRI

- Automatic cascading of deletes and updates from PK to FK tables
- ANSI standard restrict and cascade semantics

```
CREATE TABLE country (
   country_name NVARCHAR(75) NOT NULL PRIMARY KEY )

CREATE TABLE employee (
   employee_name NVARCHAR(75) NOT NULL,
   nationality NVARCHAR(75) NOT NULL REFERENCES country
   ON UPDATE CASCADE
   ON DELETE NO ACTION,
   passport_number VARCHAR(25) NOT NULL,
   PRIMARY KEY (nationality, passport_number))

CREATE TABLE renewal_reminder (
   nationality NVARCHAR(75) NOT NULL,
   passport_number VARCHAR(25) NOT NULL,
   date_of_expiry datetime NOT NULL,
   FOREIGN KEY (nationality, passport_number) REFERENCES employee
   ON UPDATE CASCADE
   ON DELETE CASCADE )
```

User Defined Functions

- Multi-statement T-SQL routines
- Scalar-valued:
  - `Select f(c1) …`
  - `Select … where f2(c2)`
  - *Usable in any expression (Order By, Group By..)*
- Table-valued (also called relational):
  - `Select c1 from f(arg)…`
- Strongly typed input args with return value
  - No output parameters
- Inline relational functions
  - Effectively a parameterized view

Scalar UDF Example

```
CREATE FUNCTION ExtractNamePart(@InName varchar(100),
   @part tinyint)
RETURNS varchar(30)  AS
BEGIN
   DECLARE @offset tinyint
   SET @offset = charindex(' ', @InName)

   RETURN CASE @part
        WHEN 1 THEN substring(@InName, 1, @offset-1)
        WHEN 2 THEN substring(@InName,@offset+1,
   len(@InName))
        ELSE NULL
    END
END
```

Column Level Collations

- Multilingual apps, app hosting, & server consolidation
- Per-database collations
  - Multiple apps with different collations
- Per-column collations
  - Deeper multi-lingual applications
- Attach & restore databases with different collations from

Column Level Collations Examples

```
--Stmt 1:Database-level collation:
CREATE DATABASE MyDB COLLATE Latin1_General_CS_AI

--Stmt 2:Use 3 different collations in single table:
CREATE TABLE Products (ProductId char(20),
     ProductName char(20) COLLATE French_CS_AS,
     Description char(20) COLLATE French_CI_AI)

--Stmt 3:Use default collation (French_CS_AS)
SELECT * FROM PRODUCTS ORDER BY ProductName, Description

--Stmt 4:Statement collation specification (French_CI_AI)
SELECT * FROM PRODUCTS ORDER BY ProductName COLLATE
     French_CI_AI

--Stmt 5:Column collation specification (French_CI_AI)
SELECT productId, ProductName COLLATE French_CI_AI
     productName, description      FROM Products
```

Instead Of Triggers

- Application
  - Allow any view to be updateable
  - Implement before triggers
- Trigger executed instead of insert, delete, or update
- Supported on view or table
- Inserted/deleted tables available
- Statement level semantics

Instead of Trigger Example

- Updateable Partitioned View
  - Partitioning Column : Region

```
CREATE VIEW CustomersAll AS
   SELECT CustomerID, CompanyName, Address, Region
   FROM CustomerEast
UNION ALL
   SELECT CustomerID, CompanyName, Address, Region
   FROM CustomerCentral
UNION ALL
   SELECT CustomerID, CompanyName, Address, Region
   FROM CustomerWest
```

Instead of Trigger Example

```
CREATE TRIGGER IO_Trig_INS_CustomersAll ON CustomersAll
INSTEAD OF INSERT AS
BEGIN
    INSERT INTO CustomersEast
           SELECT  CustomerID,CompanyName,Address,Region
           FROM inserted WHERE Region = 'East'

    INSERT INTO CustomersCentral
           SELECT CustomerID, CompanyName, Address,Region
           FROM inserted WHERE Region = 'Central'

    INSERT INTO CustomersWest
           SELECT CustomerID, CompanyName, Address,Region
           FROM inserted WHERE Region = 'West'
END --trigger action
```

New Datatypes

- Table Type
  - Return type for table valued UDFs
  - Allows easier programming of iterative operations
    - E.g. transitive closure computation
- BigInt
  - 8 byte integer
- SQL_Variant
  - Can store any base type (except LOB)
  - Can be used to implement open schema

Full Text Enhancements

- Change tracking
  - Maintain list of changes to indexed data
- Image filtering
  - Support MS Office documents & HTML
  - Custom filters (3rd party)
- Failover clustering
- Column level linguistic analysis
  - Columns can have associated language
- Top-N-By-Rank
  - Containstable ( ), Freetexttable ( )

Extended Properties

- Associate user-defined metadata with SQL objects
  - Allows easy support of Rich UI & client-side validation
- Objects → (property name, value)
- Functions:
  - Sp_setextendedproperty
  - Sp_dropextendedproperty
  - Fn_listextendedproperty
- Example :

```
exec sp_addextendedproperty 'caption',
    'Employee_Id', 'user', dbo, 'view', foo_view,
    'column', id

select * from ::fn_listextendedproperty ('caption',
    'user', 'dbo', 'view', 'foo_view', 'column',
    NULL)
```

- Properties retained on object rename

Other Developer Features

- checkSum
  - Intrinsic for computing a checksum over a set of columns
- Session context
  - Middle-tier apps can set/get user supplied context
- Tools support
  - Profiler (larger workloads, performance analysis, OLAP events)
  - Query Analyzer
  - Index Tuning Wizard (Indexed

SQL Server 2000: Relational Engine

- Programmability
- XML support
- Manageability
  - Multiple SQL Server instances
  - Security
- Scalability & performance
- Availability

Relational to XML

- SELECT … FOR XML
  - FOR XML RAW (return an XML rowset)
  - FOR XML AUTO (exploit RI, name matching, etc.)
  - FOR XML EXPLICIT (maximal control)
- Annotated Schema
  - Mapping between XML and relational schema expressed in XML
- Templates
  - Encapsulated parameterized query
  - XSL/T support
  - XPATH support
- Direct URL access (SQL “owned” virtual root)
  - SELECT … FOR XML
  - Annotated schema
  - Templates

XML to Relational

- Bulk load using XML
- Updategram
  - web delivery post SQL Server 2000
- Templates and Annotated Schema
- SQL server hosted XML tree
  - Directly insert document into SQL Server hosted XML tree
  - Select from server hosted XML tree rowset & insert into SQL tables

XML Example

```
http://SRV1/nwind?
    sql=SELECT+DISTINCT+ContactTitle+FROM+
    Customers+WHERE+ContactTitle+LIKE+'Sa
    %25'+ORDER+bY+ContactTitle+FOR+XML+AUT
    O
```

Result set:

```
<Customers ContactTitle="Sales Agent" />
<Customers ContactTitle="Sales Associate" />
<Customers ContactTitle="Sales Manager" />
<Customers ContactTitle="Sales Representative" />
```

SQL Server 2000: Relational Engine

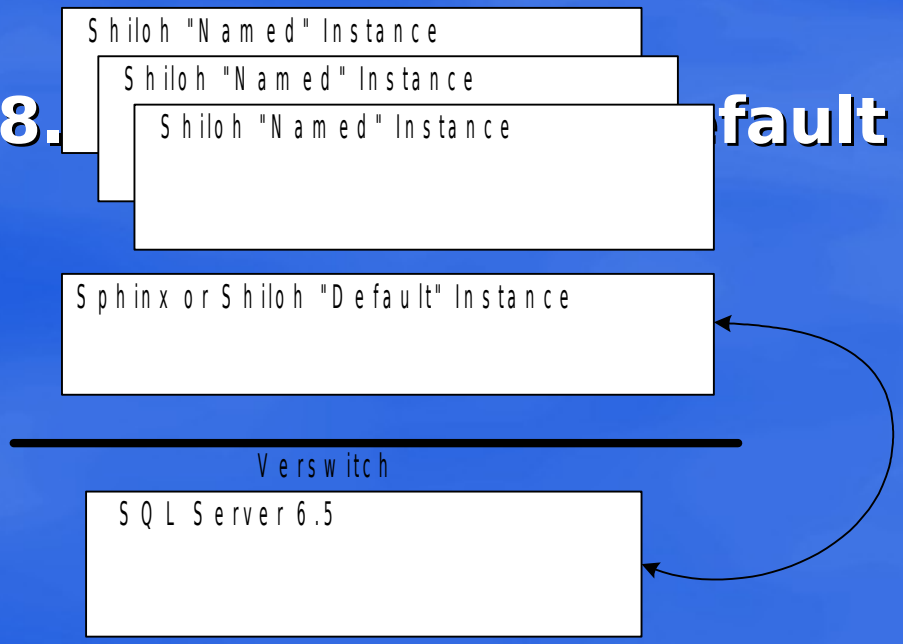
- Programmability
- XML support
- Manageability
  - Multiple SQL Server instances
  - Security
- Scalability & performance
- Availability

Multiple SQL Server Instances

- Support multiple SQL server instances on single machine
- Applications:
  - Application hosting
  - Server consolidation
  - SQL runtime (MSDE)
  - Used internally in implementation of shared disk failover

Multi-instance

- One default instance (typically server name)
- Multiple named instances (server_name\instance)
- Named instances are all version 8.0 or above
- 7.0 or 8.[illegible]efault instance



SQL Server 2000: Relational Engine

- Programmability
- XML support
- Manageability
  - Multiple SQL Server instances
  - Security
- Scalability & performance
- Availability

Security

- Default SQL Server 2000 setup is secure
- Auditing
- Encryption
  - Logins (SSL)
  - New built in function calling Win2k crypto API
- Win2K Kerberos support:
  - Linked server
- Enhancements to server roles
- C2 security evaluation

SQL Server 2000: Relational Engine

- Programmability
- XML support
- Manageability
  - Multiple SQL Server instances
  - Security
- Scalability & performance
- Availability

Partitioned Views

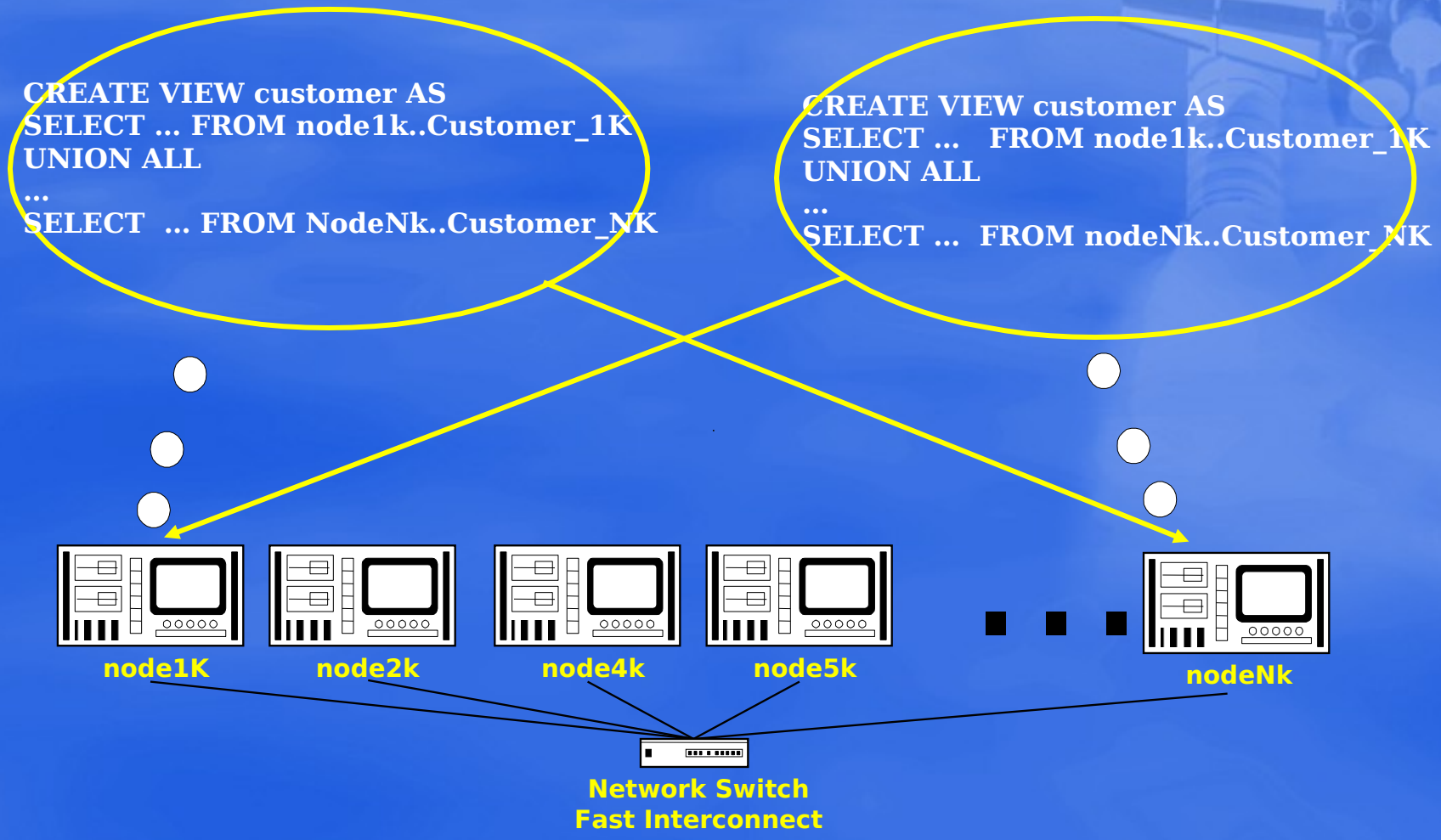
- **Partitioned view:**
  - **UNION ALL view over two or more tables**
  - **Partitioning columns with disjoint intervals**

Example:

```
CREATE TABLE CUSTOMER_1K (…, C_W_ID INT NOT NULL
   CHECK (C_W_ID BETWEEN 1 AND 1000), PRIMARY KEY(C_W_ID, …)

…
CREATE TABLE CUSTOMER_10K (…, C_W_ID INT NOT NULL
   CHECK (C_W_ID BETWEEN 9001 AND 10000),
      PRIMARY KEY (C_W_ID, …)


CREATE VIEW CUSTOMER AS
   SELECT …, C_W_ID FROM CUSTOMER_1K  UNION ALL …
```

Distributed Partitioned Views

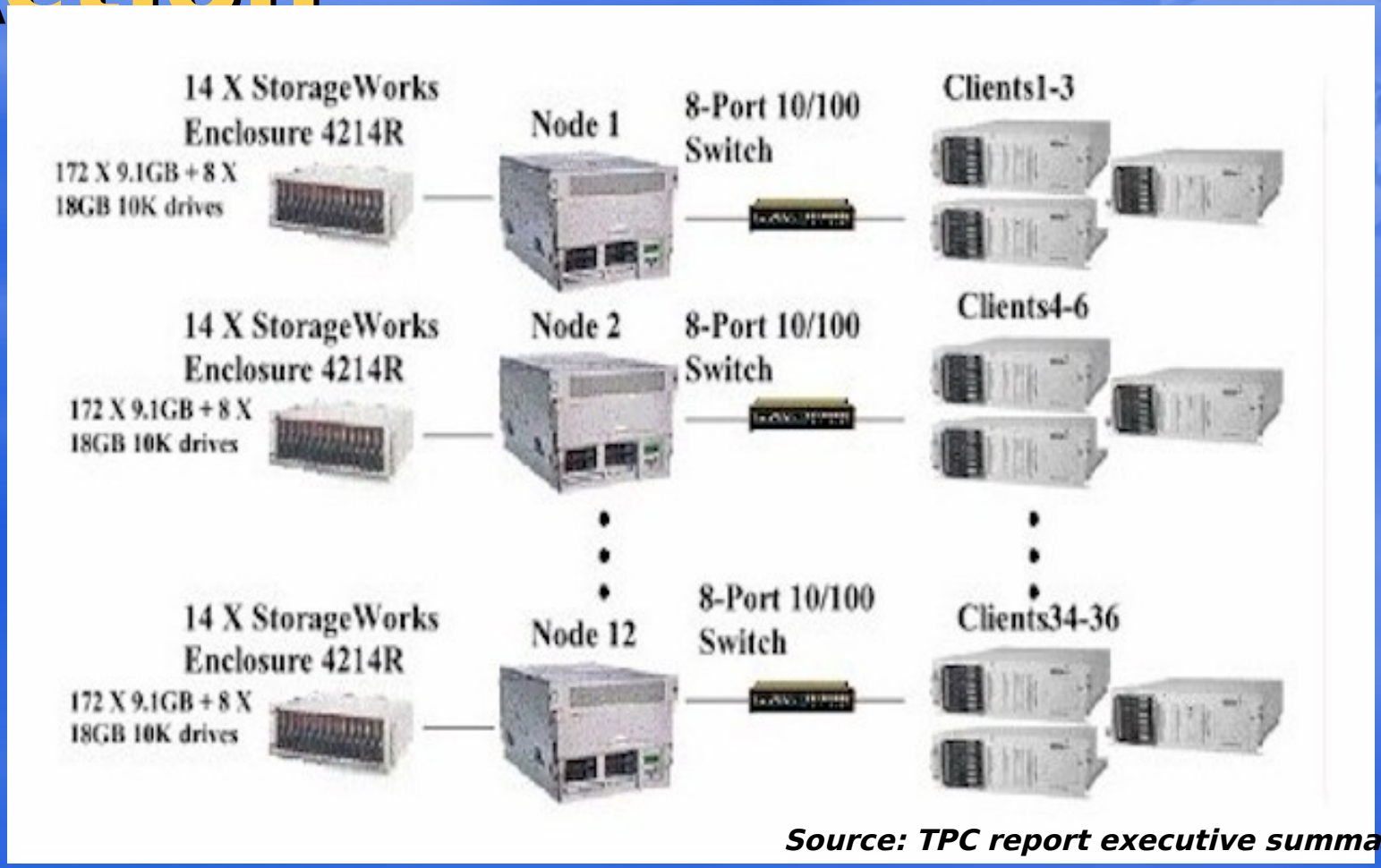
Each node has common view of tables across all nodes in virtual cluster



Distributed Partitioned Views

1. Each table in the application is split across the virtual cluster
2. Each table is partitioned into disjoint intervals using a CHECK constraint
3. Nodes are linked using OLEDB linked servers:
   - `sp_addlinkedserver 'node1'`
   - `sp_setnetname 'node1', 'MACHINE_NAME'`
   - `sp_serveroption 'node1', 'lazyschemavalidation', 'true'`
4. Create view for each table on each node linking together partitions

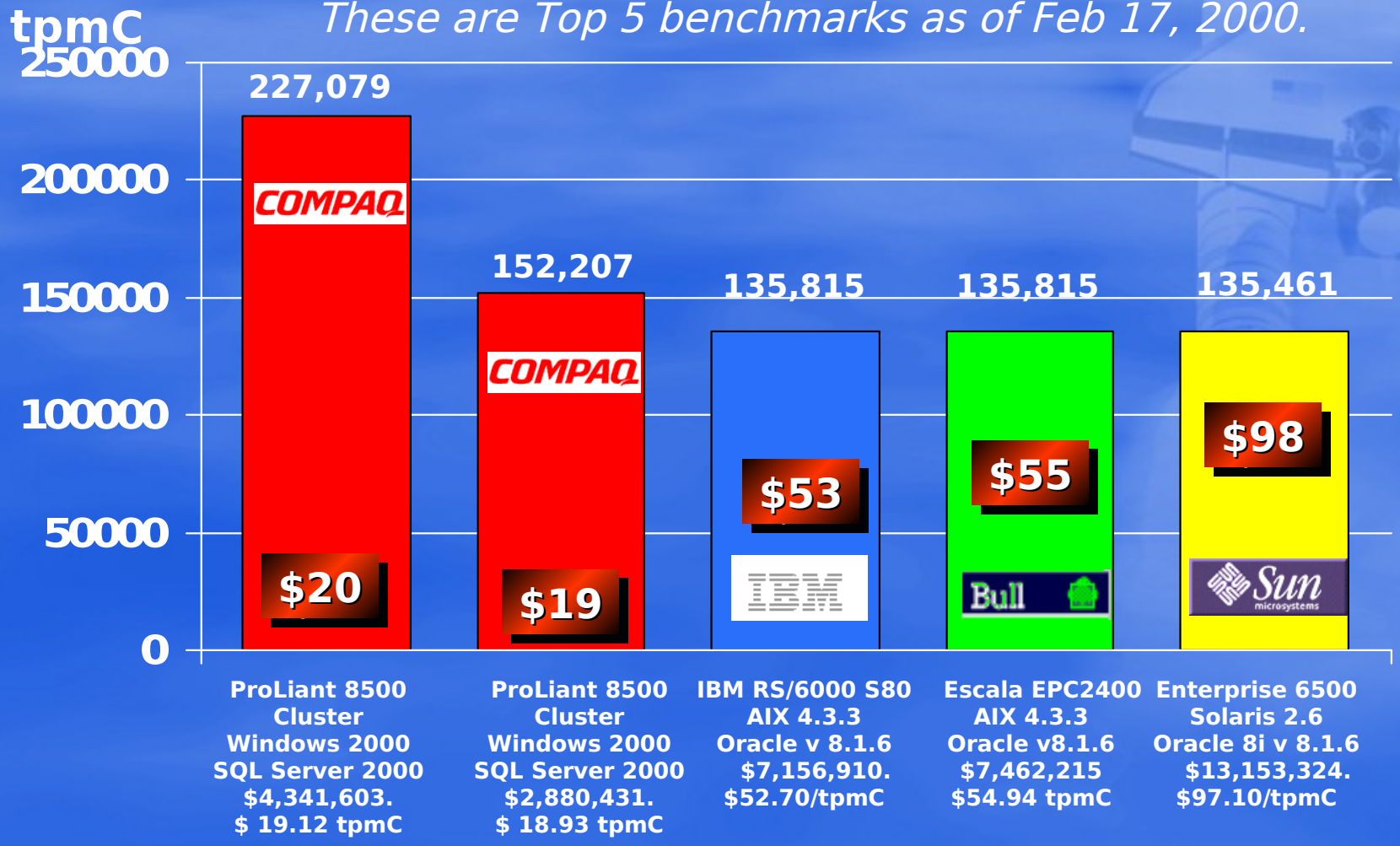
Partitioned Views in Action



Source: TPC report executive summary

Partitioned Views Scale!

These are Top 5 benchmarks as of Feb 17, 2000.



NOTE: All TPC-C results reported as of February 17, 2000

Performance

- Index enhancements
  - Indexed views
  - Other index improvements
- Parallelism
- Query optimizer enhancements
- Distributed queries
- Large memory support

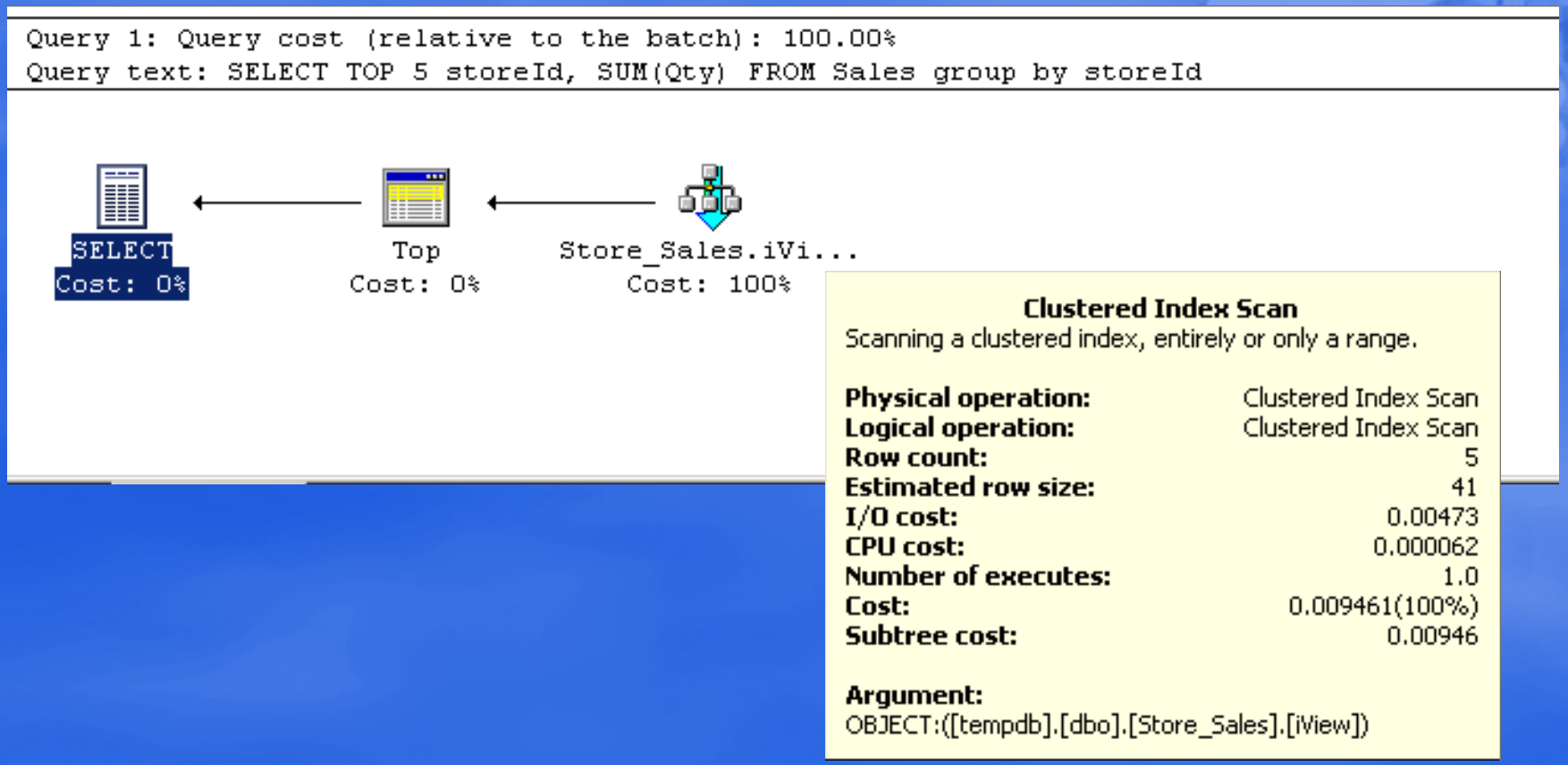
Indexed Views

- Decision support workloads
- Content of view is persisted & maintained
- View may contain joins & aggregations
  - With some restrictions
- Allows storage of partial results
- Automatically used by optimizer

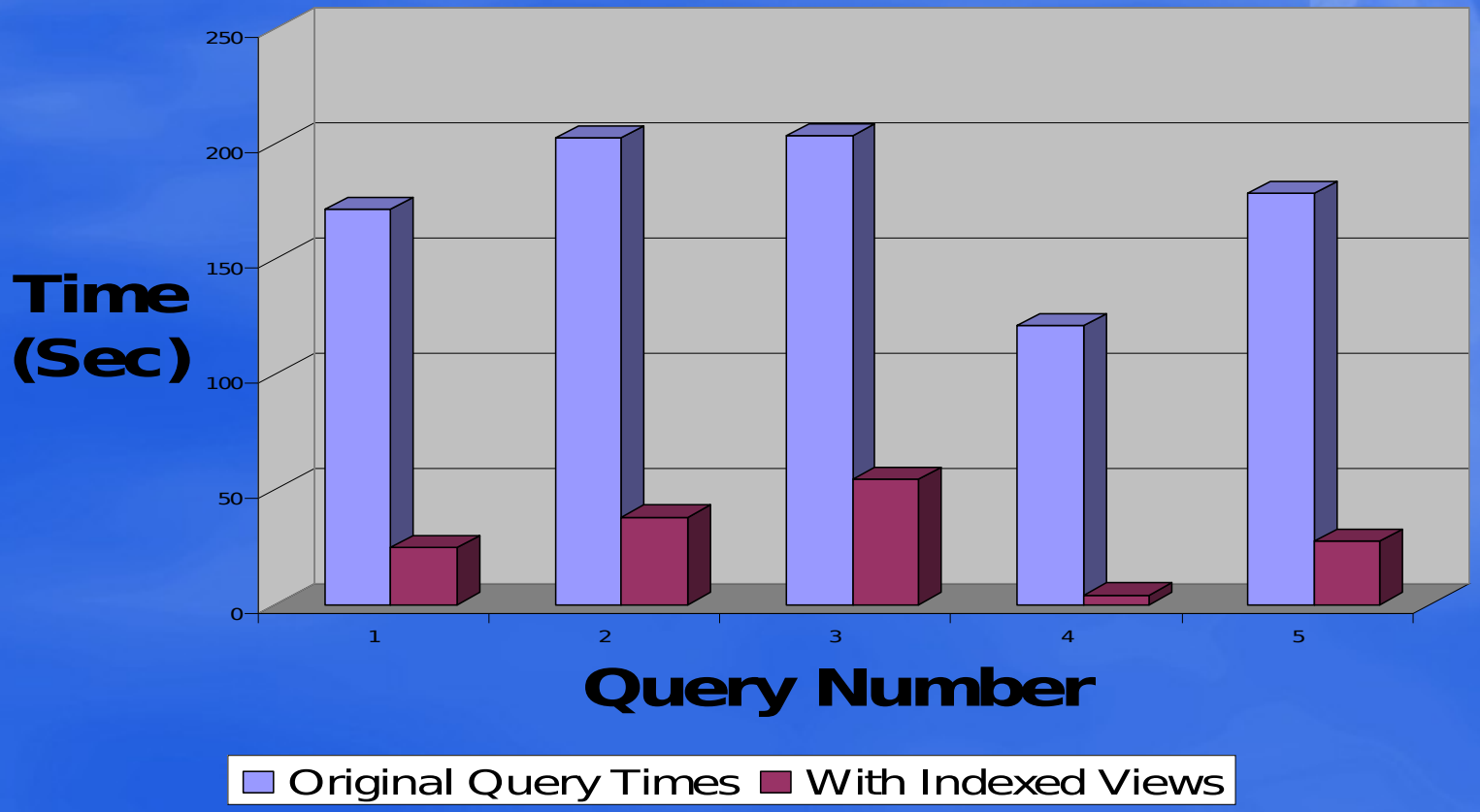
Indexed Views Example

```
--Stmt 1: Create populate table
create table sales (storeID int, qty integer not null,
    other_data varchar(20))

--Stmt 2:Create view
CREATE VIEW Store_Sales WITH SCHEMABINDING AS
    SELECT StoreId, SUM(qty) Total, COUNT_BIG(*) count
    FROM dbo.Sales
    GROUP BY StoreId

--Stmt 3: Create index on view
CREATE UNIQUE CLUSTERED INDEX iView
    ON Store_Sales(StoreId)

--Stmt 4: Select from base table (will use
    indexed view)
```

Example Query Plan



Indexes on Views



Index Enhancements

- Indexes on computed columns
  - Including PK & unique constraints
- Indexes with ASC & DESC columns
- Example S:

```
CREATE INDEX I1 ON T (a ASC, b DESC)

SELECT a, b, c FROM T WHERE a>10
   ORDER BY a ASC, b DESC
```

- RE creates & maintains indexes
- Parallel index create
- Indexes supported on bit columns

Parallelism

- Enhancements to query parallelism
  - More efficient plans
  - Bitmap filtering
  - Multi-threaded access to spools
  - Reduced parallel sort stalls
  - Elapsed time costing
  - Integrated deadlock detection
- Parallel insert/update/delete
- Parallel index creation
- Parallel DBCC

Query Optimizer

- Improved cardinality estimation
  - Exploit new statistics
    - captures spikes, skews, & frequent values
  - Improve many computations
- More granular statistics
  - Dynamic bucket boundaries
- Star joins
- Snowflake schema
- More trivial plan cases

Distributed Queries

- New Distributed query providers:
  - Active Directory
  - Exchange 2000
- New OLEDB Interfaces:
  - Statistics
  - Constraints
- Cost based optimization:
  - Obtains statistics from remote source
  - Smarter remoting logic
- More SQL dialects
  - Minimal SQL
  - More granular capabilities
- Heterogeneous partitioned views

Large Memory Support

- PSE36 (Page Size Extensions):
  - Supported in 7.0
  - Device driver providing access to up 64 gig
  - Copy-in/copy-out model
- PAE (Physical Address Extensions):
  - Direct 64 gig physical memory support in Win2K VMM
  - Exploit via multiple SQL Server instances
- VLM (Very Large Memory):
  - Supported in SQL Server 2000
  - Map-in/Map-out model addressing 64 gig
- Full 64 bit support
  - Flat 64 bit address space (no copying & no mapping)
  - Prototype running in the development shop
  - Closely tied to Win2K 64 bit release schedule

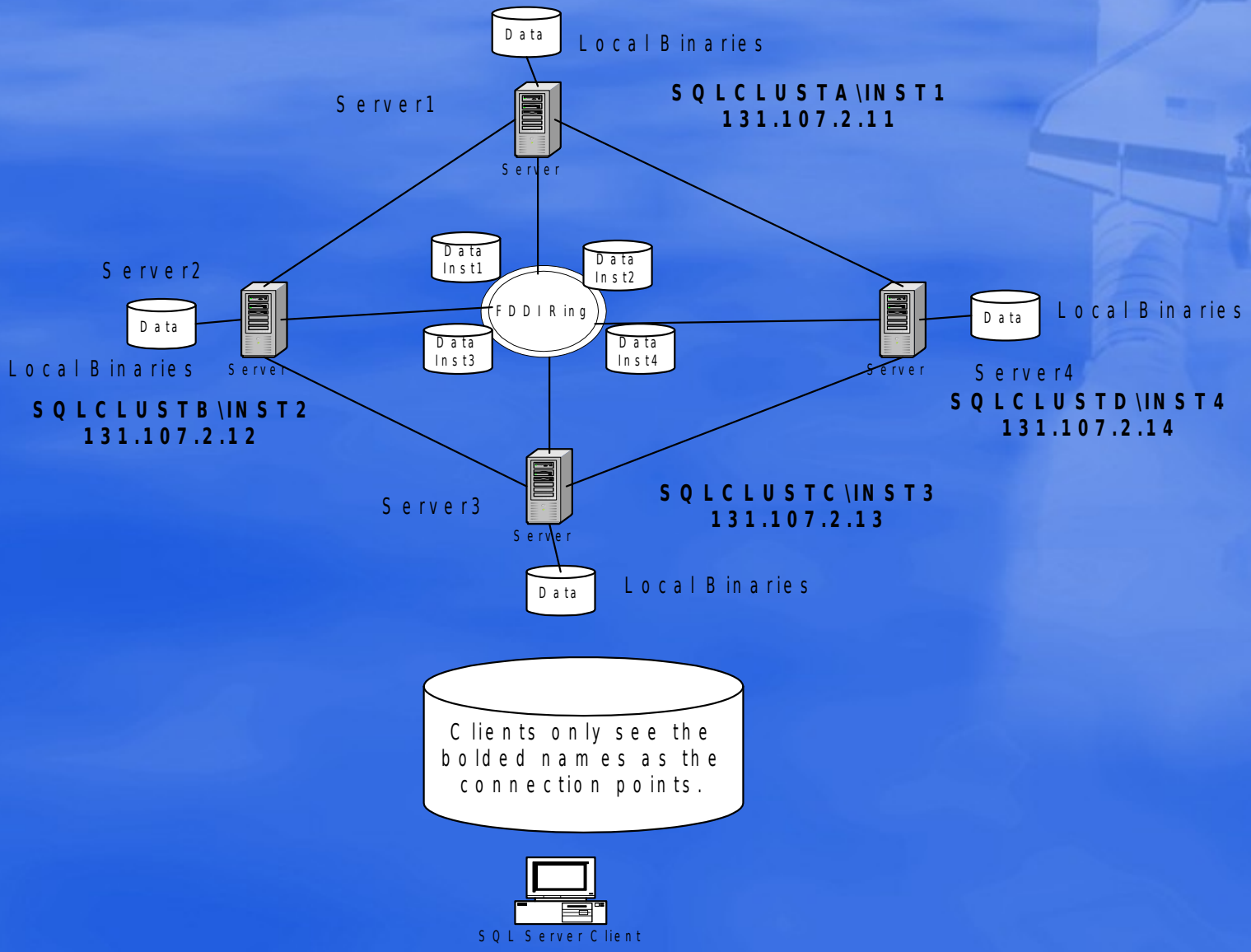
SQL Server 2000: Relational Engine

- Programmability
- XML support
- Manageability
  - Multiple SQL Server instances
  - Security
- Scalability & performance
- Availability

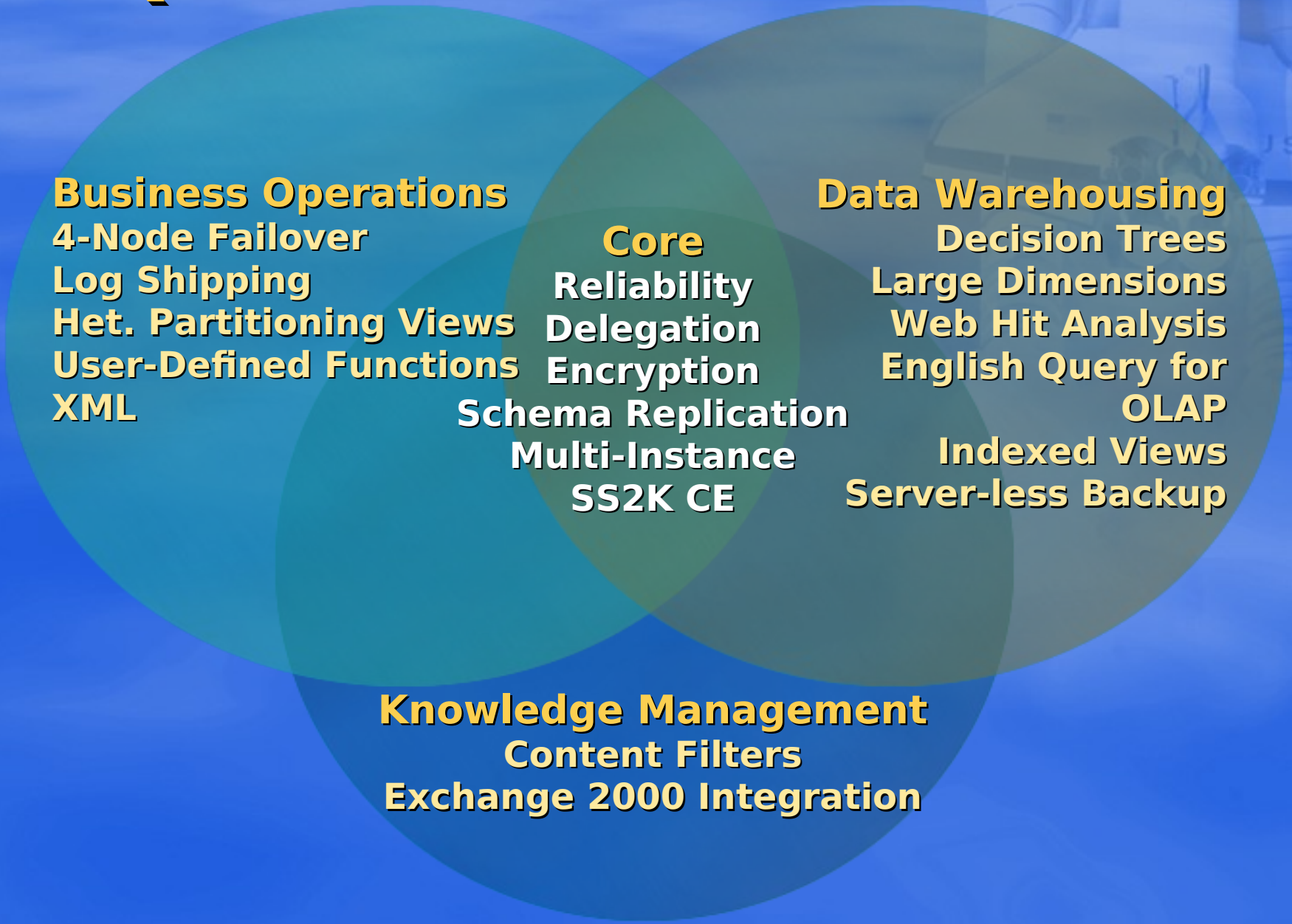
Availability

- Complete re-architecture of 7.0:
  - Uses multiple instance infrastructure
- Enhancements
  - Rolling operating system upgrades
  - Full-text indexing support
  - Disaster recovery
  - Windows 2000 Data Center support

Availability



SQL Server 2000



Questions/Feedback

POWER UP

Microsoft